Kodi: Conceptual Architecture Analysis

Link to Video

ROLES

Daniel Madan - Team Leader/External Interfaces/Use Cases

Daniel Solomon - Presenter/Architecture/Diagrams/Conclusion

Ethan Fighel - Presenter/External Interfaces/Use Cases

Zachary Kizell - Architecture/Diagrams

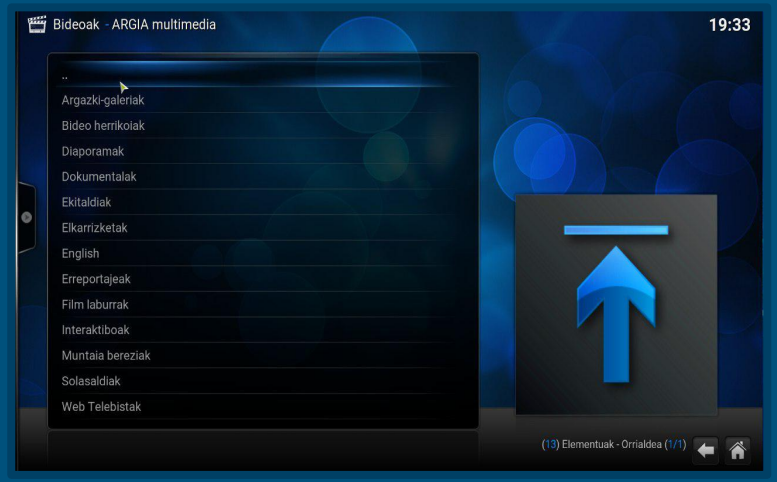
Brian Cabingan - Architecture/Diagrams

Brett Morris - Conclusions/Lessons Learned/Abstract/Introduction

Abstract

I. Kodi: An Open-Source Multimedia Center Evolution
II. Layered Architecture: Kodi's Design Approach
III. Four Key Layers: Client, Presentation, Business, Data
IV. Efficient Management for Diverse Functions
V. Flexibility for Expanding Features and Extensions

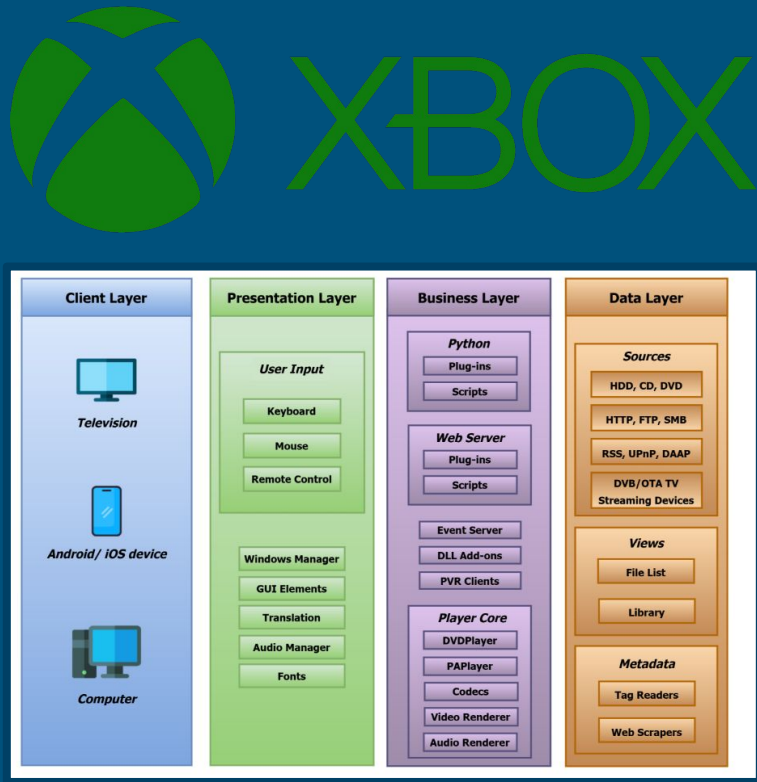
Table of Contents



Introduction

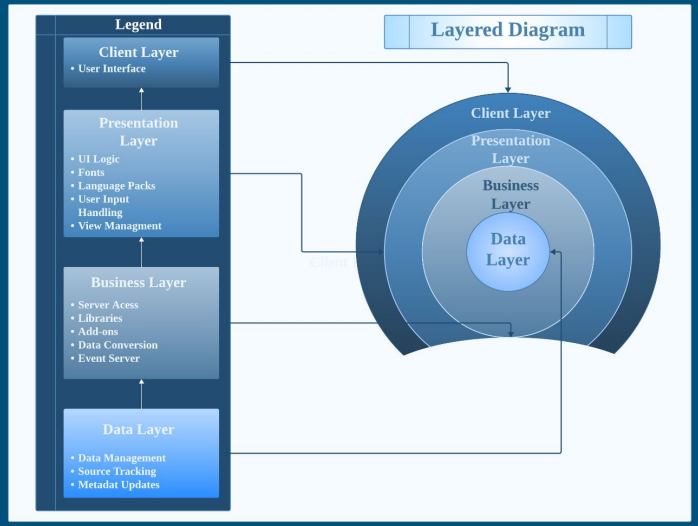
What is Kodi?

I. Kodi's versatile and open-source nature
II. Evolution from Xbox Media Player to a multimedia solution
III. Exploration of Kodi's Layered Architecture
IV. Four layers: Client, Presentation, Business, Data
V. Independent management of application aspects
VI. Facilitation of feature and extension additions



Architecture - Parts/Components

I. Kodi uses Layered Architecture Style
II. Suitable for hierarchical organization of classes or services
III. Four layers in Kodi: Client, Presentation, Business, Data
    A. Client Layer for user access on various devices
    B. Presentation Layer manages front-end interaction
    C. Business Layer handles external content, conversions, and EventServer
    D. Data Layer tracks and updates sources, files, and metadata



Parts/Components Cont’d

I. Video: Kodi's video library uses file-based metadata and FFmpeg-based DVDPlayer for playback.
II. Music: Kodi's music library enables playlist creation from music file data.
III. Media Formats: Kodi handles various media formats, including DVDs, hard drives, network file systems, and online videos.
IV. Pictures: Kodi allows photo viewing with the CxImage library for image handling.
V. Live Television: Kodi offers Live TV with EPG and DVR features, supporting various PVR backends.

Architecture - System Evolution

I. Kodi is a community-driven open-source project.
II. Developers use GitHub Issues for bug reporting and tracking.
III. Nightly builds are used for testing, followed by public betas and release candidates.
IV. Kodi's layered architecture simplifies system evolution and updates.
V. The modular structure enhances maintenance and scalability.

System Evolution Cont’d

I. Kodi's origin can be traced back to Xbox Media Player, created by Duo Egaq and Albert Griscti-Soler.
II. Source code for beta 6 was released in 2002
III. Frodo joined the team, merging Yet Another Media Player to create Xbox Media Player 2.0 in December 2002.
IV. Introduced embedded Python code support and expanded media compatibility.
V. Kodi was renamed in July 2014

Architecture - Development Team

I. Not entirely flat despite being open source.
II. A group of recognized developers reviews and approves or rejects pull requests.
III. Collaboration with users who report bugs and propose features is essential.
IV. Kodi has a board of elected members, often former developers.
V. Board members oversee task management and the development process while remaining active in coding.

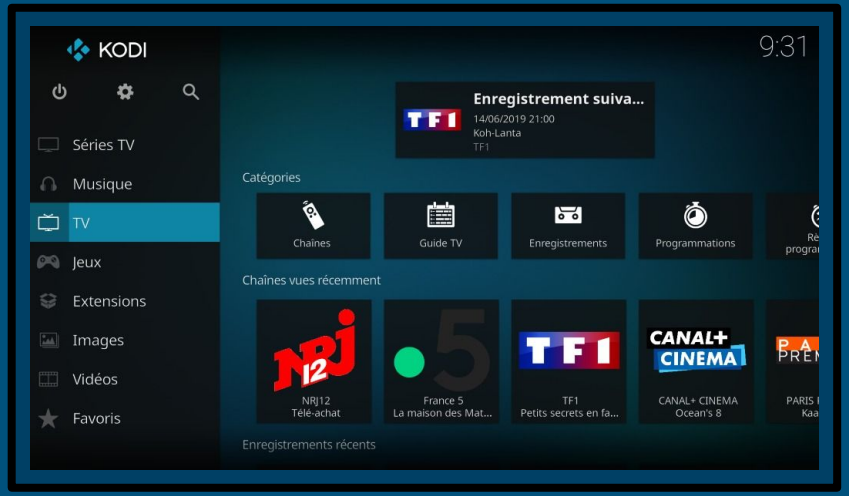
Kodi Development Team 2018 Devcon

External Interfaces

- Presentation Layer:
  - User Input Module: user input -> plugins
  - Windows Manager Module: OS libraries -> windows management
  - GUI Elements: User <-> GUI options
  - Translation Module: User Input-> Language Server
  - Audio Manager Module: User -> Kodi -> audio device
  - Fonts Module: User Input-> Font Settings

External Interfaces

- Business Layer:
  - Python Module: User -> Python Addons
  - Web Server Module: Users -> Media Content
  - Player Core Module: Kodi -> Playback Hardware/Software
- Data Layer:
  - Sources Elements: User -> Content Sources
  - Views Elements: Files -> User
  - Metadata Elements: Content Sources -> Metadata



Use Cases

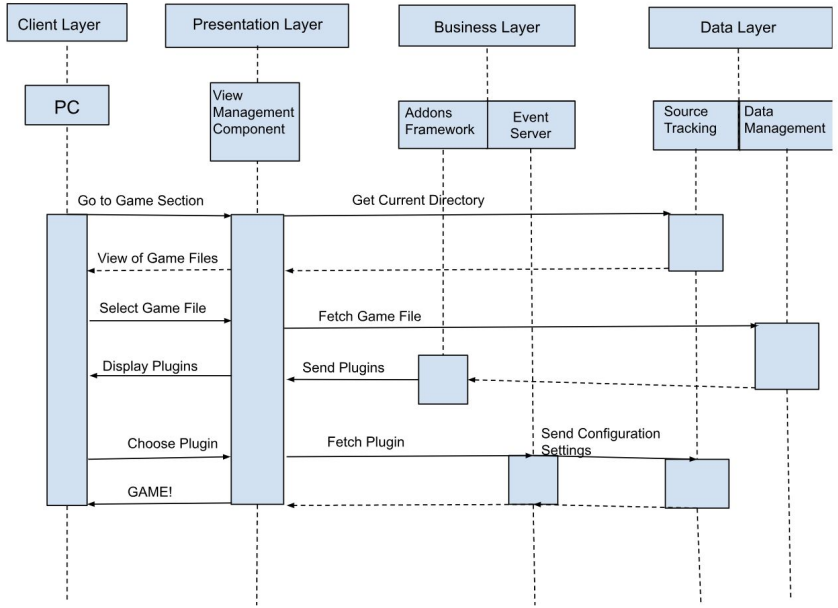


Figure 1: Sequence diagram of use case 1

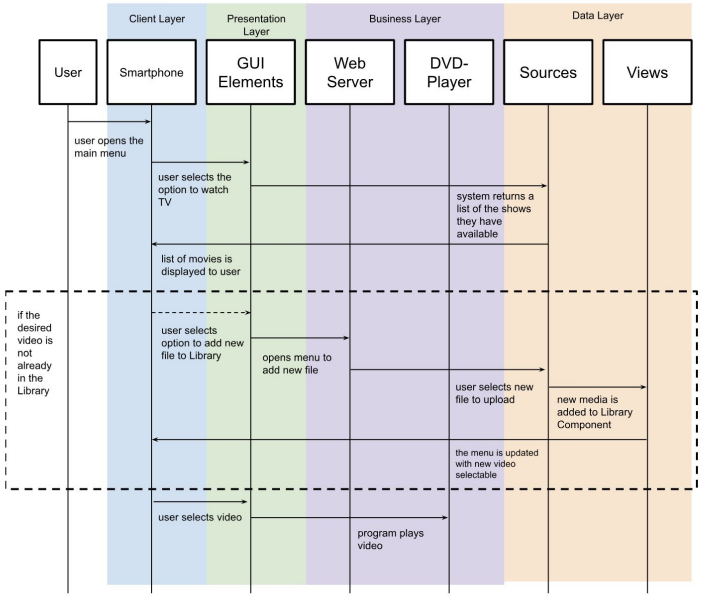


Figure 2: Sequence diagram of use case 2

Use Case 1

As a user, I would like to be able to access game files in my local file system on my PC so that I can play those games using the plugins I can download onto Kodi from the Internet

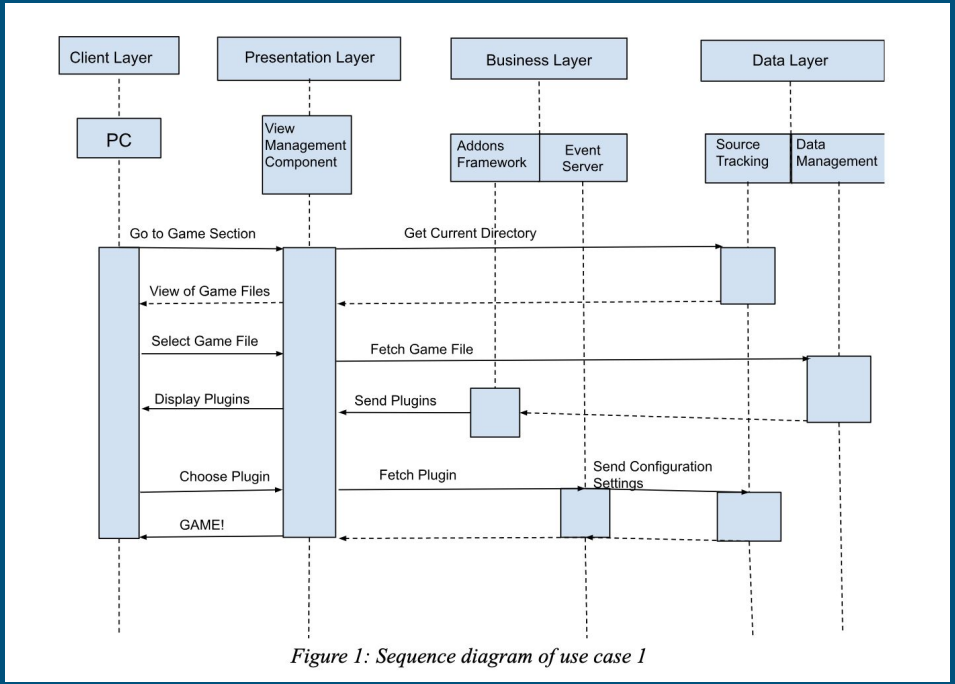


Figure 1: Sequence diagram of use case 1

Use Case 2

A user wants to search for a specific movie and watch it on their smartphone.

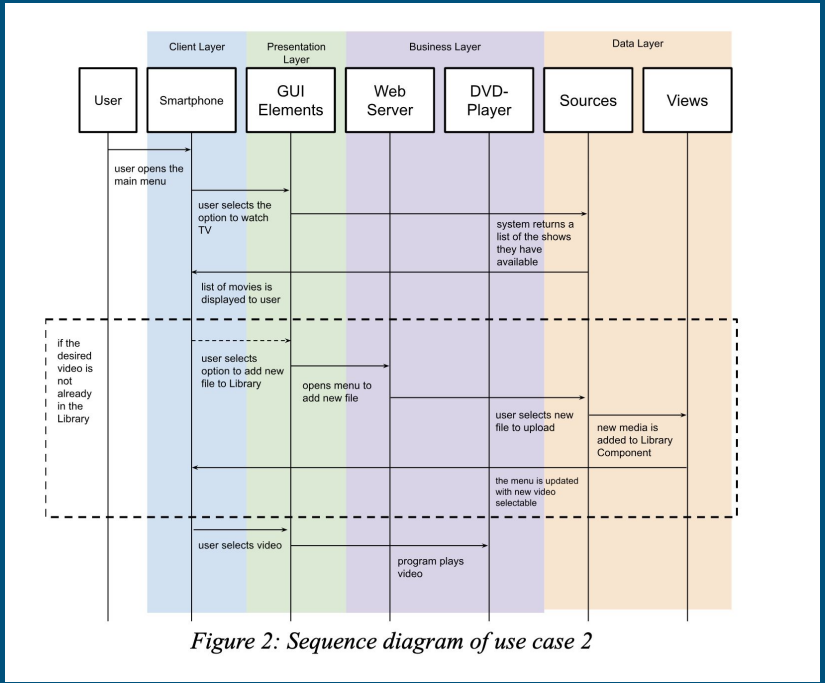


Figure 2: Sequence diagram of use case 2

Conclusion

I. Kodi's layered architecture, comprising Client, Presentation, Business, and Data layers
II. The modular design of Kodi allows for flexibility, ease of maintenance, and continuous evolution.
III. Kodi's transition from Xbox Media Player to Kodi 14 highlights growth and community development.
IV. Success emphasizes the importance of adaptable architecture and community-driven development in digital media.

Lessons Learned

I. Kodi's Layered Architecture efficiently manages multimedia applications with four layers.
II. Open-source nature encourages extensibility, ease of maintenance, and scalability.
III. Kodi's evolution showcases robust architecture and community collaboration.
IV. External interfaces, including Addons Framework and Python Scripts Interpreter, enhance versatility.
V. Support for diverse media formats, video playback, music library, image management, and live television suits various use cases.

Thank You For Listening